Noname manuscript No.
(will be inserted by the editor)

Solving Computationally Expensive Optimization Problems with CPU Time-Correlated Functions

Mark A. Abramson · Thomas J. Asaki · John E. Dennis, Jr. · Raymond Magallanez, Jr. · Matthew J. Sottile

May 27, 2008

Abstract In this paper, we characterize a new class of optimization problems in which objective function values are correlated with the computational time required to obtain these values. That is, as the optimal solution is approached, the computational time required to compute an objective function values decreases significantly. This is motivated by an application in which each objective function evaluation requires both a numerical fluid dynamics simulation and an image registration process, and the goal is to find the parameter values of a predetermined reference image by comparing the flow dynamics from the numerical simulation and the reference image through the image comparison process. In designing an approach to numerically solve the more general class of problems in an efficient way, we make use of surrogates based on CPU times of previously evaluated points, rather than their function values, all within the SEARCH step framework of mesh adaptive direct search algorithms. Because of the expected CPU time correlation, a time cutoff parameter was added to the objective function evaluation to allow its termination during the comparison process if the computational time exceeds a specified threshold. The approach was tested using the NOMADm and DACE MATLAB® software packages, and results are presented.

Mark A. Abramson
The Boeing Company, PO Box 3707 Mail Code 7L-21, Seattle WA 98124-2207 USA E-mail: Abramson.Mark@gmail.com

Thomas J. Asaki
Los Alamos National Laboratory, MS D413, Los Alamos, New Mexico 87545 USA E-mail: asaki@lanl.gov

Raymond Magallanez, Jr.
Headquarters, United States Air Force, AF/A9R, Pentagon, Washington, DC USA E-mail: Raymond.MagallanezJr@pentagon.af.mil

John E. Dennis, Jr.
Rice University, Department of Computational and Applied Mathematics, 8419 42nd Avenue SW, Seattle, WA 98136-2360 USA E-mail: dennis@rice.edu

Matthew J. Sottile University of Oregon, Department of Computer and Information Science, Eugene, OR 97401 E-mail: matt@cs.uoregon.edu

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
27 MAY 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Solving Computationally Expensive Optimization Problems with CPU Time-Correlated Functions	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Rice University,Department of Computational and Applied Mathematics,Seattle,WA,98136-2360**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this paper, we characterize a new class of optimization problems in which objective function values are correlated with the computational time required to obtain these values. That is, as the optimal solution is approached, the computational time required to compute an objective function values decreases significantly. This is motivated by an application in which each objective function evaluation requires both a numerical fluid dynamics simulation and an image registration process, and the goal is to find the parameter values of a predetermined reference image by comparing the flow dynamics from the numerical simulation and the reference image through the image comparison process. In designing an approach to numerically solve the more general class of problems in an efficient way, we make use of surrogates based on CPU times of previously evaluated points, rather than their function values, all within the search step framework of mesh adaptive direct search algorithms. Because of the expected CPU time correlation, a time cutoff parameter was added to the objective function evaluation to allow its termination during the comparison process if the computational time exceeds a specified threshold. The approach was tested using the NOMADm and DACE MATLABr software packages, and results are presented.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**11**	

Keywords Derivative-free optimization, mesh adaptive direct search (MADS) algorithms, pattern search, Navier-Stokes equations, surrogate optimization, image registration

1 Introduction

In this paper, we introduce a new class of optimization problems and a novel approach for solving them. This class consists of minimizing an objective function $f : X \subset \mathbb{R}^n \to \mathbb{R}$ that is computationally expensive to evaluate, but becomes significantly less so as the solution is approached. Typically, the objective function involves some type of realistic engineering simulation, which in practice can exceed days or weeks of wall clock time on even the largest parallel systems, making it infeasible to naively perform full simulation runs on a large set of problem instances. While tackling the extreme computational requirements of these large simulations is our ultimate goal, our primary motivation in this paper comes from an application that involves both a fluid dynamics simulation and an image registration process, the latter of which becomes much less expensive close to the solution.

This unique property of correlation between objective function values and the computational time required to compute them leads us to a solution approach that integrates CPU runtime measures into the optimization process, allowing us to better utilize computational resources and significantly reduce computational time. Because of the computational expense, our approach also involves the iterative use of surrogates. In this context, a surrogate can be thought of as a much less expensive replacement for, but not necessarily a good approximation to, the objective function.

To maintain rigorous convergence properties of the approach, the use of surrogates is incorporated into the SEARCH step of the class of mesh adaptive direct search (MADS) algorithms. MADS was introduced by Audet and Dennis [6] as a way of extending generalized pattern search (GPS) [4,11,12,23] to optimization problems with nonlinear constraints without the use of penalty parameters [13] or filters [5]. Like GPS, each iteration of MADS consists of a SEARCH and a POLL step. The SEARCH step, which consists of evaluating the objective function at any finite number of points, allows for the use of surrogates to handle computationally expensive functions. The POLL step, which consists of evaluating adjacent points on a mesh formed by directions that positively span $\mathbb{R}^n$, drives the convergence theory of the algorithm. Since our solution approach focuses only on a carefully designed SEARCH step within the MADS framework, and without disturbing the algorithm or its theoretical convergence properties, we restrict our discussion only to the SEARCH step. Further details on the MADS algorithm class and its convergence properties can be found in [6] or [2].

The remainder of the paper is as follows. In Section 2, we further motivate our work by discussing the details of our application. In 3, we discuss surrogates in more detail, including some specific types and initialization strategies. In Section 4, we introduce new strategies for incorporating surrogates to efficiently solve our target class of problems. Numerical results on a specific instance of our application are given in Section 5, followed by some concluding remarks in Section 6.

2 An Applicable Class of Optimization Problems

The class of problems we target is motivated by an application in which each objective function evaluation requires both a fluid dynamics simulation and an image registration. The movement of fluids in a region $\Omega \subset \mathbb{R}^n, n \in \{2, 3\}$ is governed by the well-known Navier-Stokes equations:

$$\begin{aligned} \frac{\partial}{\partial t} v + (v \cdot \nabla) v + \nabla p &= \frac{1}{Re} \triangle v + (1 - \beta \tilde{T}) g, \\ \frac{\partial \tilde{T}}{\partial t} + v \cdot \nabla \tilde{T} &= \frac{1}{Re} \frac{1}{Pr} \triangle \tilde{T} + q''', \\ \text{div } v &= 0, \end{aligned}$$

where u is a velocity field on $\mathbb{R}^n$, p is the pressure field in Ω, g indicates body forces in Ω, $Re \in \mathbb{R}$ is the Reynolds number of the flow, $Pr \in R$ is Prandtl number of the flow, $\beta \in \mathbb{R}$ is the coefficient of thermal expansion, q''' is the heat source, $\tilde{T}$ is the temperature, and $\triangle = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$ denotes the Laplace operator.

Our test example is that of the well-known lid-driven cavity problem [9], in which an initially stationary 2d fluid in a rectangular container is subject to forces imposed by the top boundary (lid) moving at a uniform horizontal velocity. Since the Navier-Stokes equations cannot be solved analytically, they must be solved numerically using a finite element method and associated finite differencing scheme. We use the method of [9]. Figure 1(a) shows a snapshot of the fluid velocity at some positive time. Figure 1(b) shows the corresponding representation of the heat function H, which we will use as reference data. The heat function defines the 2d heat flux $\Phi_q = \nabla \times H$, and is analogous to the hydrodynamic stream function which defines the 2d velocity.

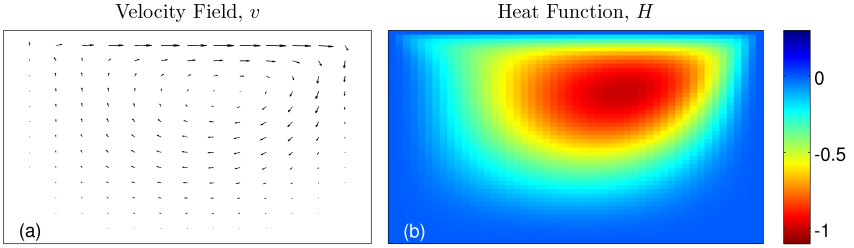

Fig. 1 Example time snapshots of a lid-driven cavity simulation. The velocity field (a) at some time $t > 0$ shows a vortical structure. The heat function (b) at time t serves as our example reference image.

Image registration is the process of estimating some optimal transformation u^* between two images. Thus, a transformation u is realized as a path through the space of images. A particular choice of u will depend on the needs of the application. For example, in medical imaging it is desirable to compare images with minimal distortion, $\nabla \times u$. Other image comparison tasks benefit from minimizing the work required to "move" the intensities from one image to another. Different types of transformations are described in [15]. If we consider the classical inner product space $L_2(\Omega)$ of squared Lebesgue-integrable functions with its standard induced norm, a transformation of an

image T is given by $T_u(x) = T(x-u(x))$, where $u(x)$ is the displacement of the point x. The objective is to minimize the distance between a *reference image* R and a *template image* T through an optimal warp transformation $T_u \in L_2(\Omega)$, as defined by some distance measurement D, and a smoothing or regularizing term S. This problem is given by

$$\min_u D[R, T_u] + \alpha S[u], \tag{1}$$

where $\alpha > 0$, and

$$D[R, T_u] = \phi(x, u(x)) \tag{2}$$

$$S[u] = A[u](x), \tag{3}$$

for a force measurement ϕ and partial differential operator A. The distance measure defines a local spatio-temporal force ϕ needed to "move" T toward R, creating the image warp transformation T_u. The regularizing term is added to the objective function to differentiate between possible transformations, since the minimum distance may not be unique, and one type of transformation may be preferred over another. Applying the Euler-Lagrange equations to (1)–(3) yields the system of nonlinear differential equations,

$$A[u](x) - \phi(x, u(x)) = 0,$$

from which the iteration scheme,

$$A[u^{k+1}](x) - \phi(x, u^k(x)) = 0, \tag{4}$$

is constructed. In particular, we chose the following, consistent with [15]:

$$D[R, T_u] = \frac{1}{2}\|T_u - R\|_{L_2(\Omega)}, \quad S[u] = \frac{1}{2}\sum_{i=1}^{n}\int_\Omega (\triangle u_i)^2 dx, \quad A[u] = \triangle^2 u.$$

Figure 2 shows an example of an image registration of two different simulated flows of heat for a fluid. The top left picture is the reference image R for a specific set of (unknown) parameter values (e.g., Reynolds number of the fluid). For a different set of parameter values, the simulation is run, resulting in the template image T shown in the top right picture. The image registration is then applied using (4) to solve (1)–(3), the solution of which is an optimal warp function $u^*(x)$. The resulting warped template image T_u and the difference between R and T_u are shown in the bottom left and right images, respectively.

Given pixel points $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$, where N is the number of pixels in the image, the objective function is a measure of dissimilarity between the images; namely,

$$d = \|u^*(x) - \bar{u}\|_2, \quad \bar{u} = \frac{1}{N}\sum_{i=1}^{N} u^*(x_i) \tag{5}$$

The subtraction of the means in (5) allows for a zero objective function value for images that are identical except for translational alignment issues. If the images are very similar, the numerical image registration scheme (4) requires only a few iterations to transform T into R, resulting in less computational time and a small distance value. On the other hand, images that are relatively dissimilar require more iterations of (4), which would increase the computational time and distance value. Thus, the objective function (distance) value and its associated computational time are expected to be strongly correlated.

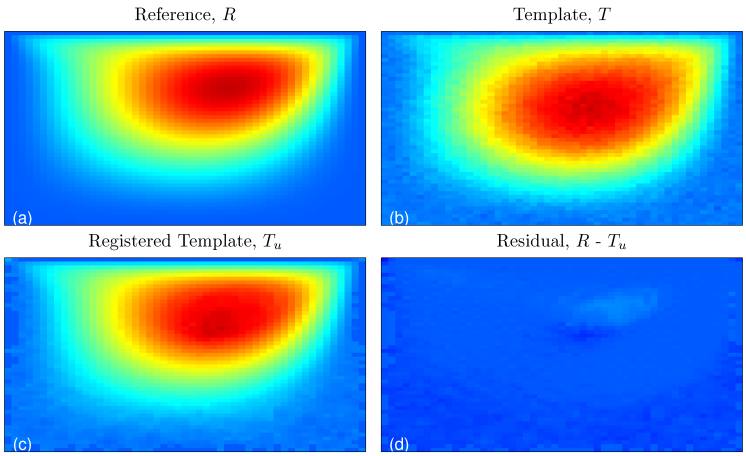

Fig. 2 Image registration example. The noisy intensities of the template image (b) are transformed into a registered image (c) intended to match a reference image (a). The optimal transformation u^* is determined by equation 1. The residual $R - T_u$ is shown in (d).

3 Surrogate Functions

The idea for surrogates first appeared as "approximation concepts" in the work of Schmit and Miura [10]. Booker et al. [8] characterize a class of problems for which surrogate functions would be an appropriate approach, suggest a surrogate composition, and set forth a general Surrogate Management Framework (SMF) for using surrogates to numerically solve optimization problems. Most surrogates are one of two types: simplified physics or response-based. A simplified physics model, also known as a low-fidelity model, makes certain physical assumptions that significantly reduce the computational cost by eliminating complex equations and even the number of variables. Although several novel approaches exist in the literature for treating this class of surrogates (e.g., see [18] or [3]), the actual construction of the models is problem-dependent.

To handle our target class of problems, we use as surrogates a class of response-based Kriging approximation models [19]. Given a set of known data points $\{s_i\}_{i=1}^m \subset \mathbb{R}^n$ and their function values or deterministic responses $y_s \in \mathbb{R}^m$ (*i.e.*, $[y_s]_i = y(s_i), i = i, 2, \ldots, m$), the deterministic function $\hat{y}(z)$ is modeled as a realization of a stochastic process,

$$Y(z) = \sum_{j=1}^{p} \beta_j \hat{f}_j(z) + Z(z) = \beta^T \hat{f}(z) + Z(z), \tag{6}$$

where $Y(z)$ is the sum of a regression model with coefficients $\beta = [\beta_1, \beta_2, \ldots, \beta_p] \in \mathbb{R}^p$ and basis functions $\hat{f} = [\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_p]$, and a random variable $Z(z)$, where $Z : \mathbb{R}^n \longrightarrow \mathbb{R}$, with mean zero and covariance $V(w, z) = \sigma^2 R(\theta, w, z)$ between $Z(w)$ and $Z(z)$, σ^2 is the process variance, and $R(\theta, w, z)$ is the correlation function of w and z. The parameter $\theta \in \mathbb{R}^n$ controls the shape of the correlation function.

Kriging produces an approximate function value at an unknown point $z \in \mathbb{R}^n$ using weights on known responses; namely,

$$\hat{y}(z) = c(x)^T y_s, \tag{7}$$

where $c(x) \in \mathbb{R}^m$ is a vector of weights obtained by minimizing the mean squared error (MSE) between the true model (6) and the approximate model (7), while forcing the predictor to be unbiased. Details for computing $c(x)$ are given in [19] or [14], for example.

The DACE process requires specification of the data sites, and regression and correlation functions. For constructing an initial surrogate, the set of initial data sites can be chosen via experimental design [20] or by sampling a set of "space-filling" points, such as Latin hypercube designs [21,22] or orthogonal arrays [17]. Although experimental designs generally require more function evaluations, we chose to use a central composite design (CCD) because the small dimension of our problem allowed us to sample enough points to form a more accurate second-order regression model.

The choice of correlation function can significantly impact the performance of the algorithm. A common choice in practice [14] is the Gaussian function, given by

$$R(\theta, a, b) = \prod_{j=1}^{n} R_j(\theta_j, |d_j|), \quad R_j(\theta_j, |d_j|) = \exp(-\theta_j d_j^2), \quad d_j = a_j - b_j \tag{8}$$

for any points $a, b \in \mathbb{R}^n$. In constructing the surrogate, the values for $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$, are computed as the solution to an optimization problem, based on previously evaluated points. The optimization process requires the computation of $R(\theta, x, x)^{-1}$. However, this is difficult numerically because R can become ill-conditioned as points cluster together during the convergence process of MADS [7]. The increase in the condition number of $R(\theta, x, x)$ (which can be estimated and monitored) can greatly impact the computed value of θ or prevent the calculation of the regression coefficients altogether. If the condition number is too large, then the SEARCH step is skipped.

4 New Surrogate Strategies

In this section, we introduce a new SEARCH step for CPU time-correlated functions. First, if objective function values and CPU times are positively correlated, then improvement in the objective function should not be expected once the computational time exceeds a certain amount. To avoid wasting unnecessary CPU time, we introduce a CPU time threshold parameter $t_k^{cut} > 0$ to allow a function evaluation to be aborted if it is taking too long to perform. A value of $t_k^{cut} = \infty$ means that the function is evaluated normally without being aborted.

Using this notation an evaluation of f can now be represented by $[z, t] = f(y, t_k^{cut})$, where t is the time needed to compute the function value z at a trial point $y \in X$ and for a specified value of t_k^{cut}. Once the time for computing the function value exceeds the value specified by t_k^{cut}, evaluation of $f(y)$ is aborted without returning a value for z (or z is set to be infinity or an arbitrarily large number). One possible approach we considered is to set $t_{k+1}^{cut} = t_k$, where t_k is the recorded CPU time for the current best iterate x_k.

The CPU time correlation also means that a surrogate based on either the objective function values or CPU times would probably be a good predictor of decrease in the

objective. In fact, a surrogate on the CPU time has the added advantage that it always returns a value, whereas, the objective function would be aborted if t_k^{cut} is exceeded at iteration k. We denote these surrogates on objective values and CPU times by $F_k(x)$ and $T_k(x)$ (at iteration k), respectively, and we consider the following four surrogate optimization problems, whose solutions we would expect to be good subsequent trial points for the true objective function:

$$\min_{x \in X} F_k(x), \tag{9}$$

$$\min_{x \in X} T_k(x), \tag{10}$$

$$\min_{x \in X} F_k(x), \quad \text{s.t.} \quad T_k(x) \leq t_k^{cut} + \varepsilon, \tag{11}$$

$$\min_{x \in X} T_k(x), \quad \text{s.t.} \quad F_k(x) \leq z_k. \tag{12}$$

The parameter $\varepsilon > 0$ added to the constraint in (11) is a constant offset to allow for variability in computational time.

The surrogate optimization problem is typically solved using a recursive call to MADS. Constraints in (11)–(12) are treated by the barrier approach of only allowing feasible points (or setting the function value at any infeasible trial point to infinity).

We should note that the combination of MADS with a barrier and the use of the t_k^{cut} in the original optimization problem causes a dilemma when using surrogate functions. Using the parameter t_k^{cut} to stop unprofitable function evaluations is good for saving computational time, but it results in infinite or arbitrarily large function values, which cannot be used to construct a good surrogate. To overcome this, we simply set the function value to the largest value seen thus far in the iteration process, whenever the time cutoff threshold is exceeded. Our algorithm can be summarized as simply MADS with the specific kth SEARCH step given in Figure 3.

- For all previously evaluated points X_k, construct surrogate functions $F_k(x)$ or $T_k(x)$.
- Solve a surrogate problem (one of (9)–(12)), yielding a set of trial points S_k.
- Evaluate points $y \in S_k$ using $[z, t] = f(y, t_k^{cut})$ until an improved mesh point has been found, or until all points in S_k have been evaluated.
- If $z < z_k$ for some $y \in S_k$,
  Set $x_{k+1} = y,\ z_{k+1} = z,\ t_{k+1} = t,$ and update t_{k+1}^{cut}.
  Set $k = k + 1$ and repeat SEARCH

  End

Fig. 3 MADS SEARCH step k for CPU time-correlated functions

5 A Numerical Example

We now present a numerical example, which is a specific example of the class of problems described in the Section 2. The problem is the well-studied *lid-driven cavity* problem [15]. For a given two-dimensional square domain, the Navier-Stokes equations describe a fluid flow with a horizontal velocity force on one edge (see Figure 1). The fluid begins at rest, and as time starts, a constant horizontal velocity is asserted along the top edge, causing a circular pattern of flow to appear within the fluid over time.

For each combination of Reynolds number and simulation length, the velocity and viscosity of the fluid form a different circular heat pattern throughout the region. At one particular Reynolds number and simulation length, a reference image of the heat pattern is captured and then noise is added to the image, so as to represent what one might to see in experimentally obtained physical measurements. As stated in Section 2, the goal is to run a simulation for different Reynolds numbers and simulation lengths, capture the template image, and compare the template and reference images of heat in an attempt to determine the original Reynolds number and simulation length set for the reference image. Figure 2 shown earlier actually shows reference and template images for this very problem.

To solve the problem, the MADS algorithm with the SEARCH step described by 3 was run using two MATLAB® software packages, NOMADm [1] for the implementation of MADS, and DACE [16] to build the surrogates, along with some custom-built files used for the SEARCH step. The surrogate optimization problem is solved by a recursive call to the NOMADm optimizer from within the SEARCH step.

For each scenario, different variations of the algorithm are applied and compared to a base case. The base case implementation uses GPS with an empty SEARCH step (*i.e.*, it is skipped), a single initial point or set of CCD points, and $t_k^{cut} = \infty$ for all k. This allows for a full evaluation of all points and a comparative analysis of the proposed algorithm. The other cases use the partial and full implementation of the SEARCH step presented in Figure 3.

We first made some preliminary runs using a strategy of setting $t_{k+1}^{cut} = t_k$ at each iteration. However, these runs turned out to be unsuccessful in forming good surrogates (*i.e.*, surrogates that routinely found good trial points to evaluate) because our main assumption of CPU time correlation turned out to only hold locally. If the template image is too dissimilar from the reference image, the image registration process actually terminates prematurely – with a lower CPU time and a much worse objective function value. Setting $t_{k+1}^{cut} = 2t_k$ at each iteration seemed to rectify the situation. We also experienced ill-conditioning of the correlation matrix as a solution is approached, which is caused by trial points becoming more clustered together. This was remedied by invoking an empty SEARCH (*i.e.*, not optimizing the surrogate) whenever the matrix became ill-conditioned. This is not unreasonable in this case because the CPU time correlation means that function values are probably much less expensive at this point in the iteration process, and the use of surrogates is then not as important.

The results for each case are shown in Table 1, where the column headings denote, respectively, the type of SEARCH step executed (Search), type of initial points used (x_0), final solution (x^*), number of iterations (nIter), number of function evaluations (nFunc), CPU minutes required (CPU), and the ratio of successful to total surrogate SEARCH steps executed (Successes). Except for the "None" designation, the SEARCH types refer to the four surrogate strategies shown in (9)–(12). The first letter indicates the objective function ($F(x)$ vs. $T(x)$), and the second (if present) indicates the constraint. The initial point types consist of using a single initial point in the geometric center of the bound constrained feasible region (center), a randomly chosen initial feasible point (random), or central composite design (CCD). The final solution is expressed as [Reynolds number, simulation length (seconds)]. The first three runs are base cases with no SEARCH step or time cutoff parameter used, while the last seven cases are different variations of the new SEARCH step, the first three being similar to the base case (no SEARCH step) except for the use of the time cutoff parameter to abort expensive function evaluations. (The random point is the same random initial point that was

used for the base case.) The final four cases employ one of the surrogate optimization problems (9)–(12), as just described.

Table 1 GPS: Lid-Driven Cavity Results

Full-Time ($t_k^{cut} = +\infty$):

Search	x_0	x^*	nIter	nFunc	CPU	Successes
None	center	[134.13, 4.76]	56	123	126.73	
None	random	[134.12, 4.76]	58	118	178.31	
None	CCD	[134.13, 4.76]	40	107	196.58	

Cut-Time ($t_k^{cut} = 2 \times t_k$):

Search	x_0	x^*	nIter	nFunc	CPU	Successes
None	center	[134.13, 4.76]	80	157	257.67	
None	random	[134.12, 4.76]	92	162	796.40	
None	CCD	[134.13, 4.76]	40	107	109.73	
F-T	CCD	[134.19, 4.76]	73	162	135.78	23/44
F	CCD	[134.19, 4.76]	72	165	182.89	15/34
T-F	CCD	[134.19, 4.76]	52	133	74.00	7/15
T	CCD	[134.19, 4.76]	49	127	74.48	5/13

All runs successfully found the optimal solution at essentially the same parameter values (with $f(x*) = 0.60$ in all cases). As hoped for, the CPU time was significantly lower for the two time-based surrogates ((10) and (12)) than all the other cases, despite costing more function evaluations than many other cases, including all three bases cases. This also indicates that for this class of problems, the number of function evaluations is not a good measure of the efficiency of the different implementations.

The results with no SEARCH illustrate the importance of finding an appropriate initial point to set up the time cutoff parameter. With only one initial point, this parameter needs several iterations to build enough slack to allow the sequence of points to overcome the local nature of the CPU time correlation. The extra iterations resulted in a different path to a solution, which required significantly more CPU time. However, when using a initial CCD (with an empty SEARCH step), the results are identical except for the CPU time. In this case, using the time cutoff parameter saved almost 90 CPU minutes.

Figure 4 further illustrates the performance of the four surrogate strategies, with each color representing a different strategy, and each shape representing the source (SEARCH or POLL step) of the improvement. The figure shows that the surrogates based on computational time achieve lower function values quicker than the surrogates based on function values. Furthermore, regardless of the surrogate objective, the use of a constraint in each case resulted in faster initial convergence than the corresponding unconstrained version.

6 Concluding Remarks

This paper represents a first attempt at numerically solving the challenging class of optimization problems in which function values and the CPU times required to compute them are correlated. Exploiting knowledge about CPU time correlation with objective

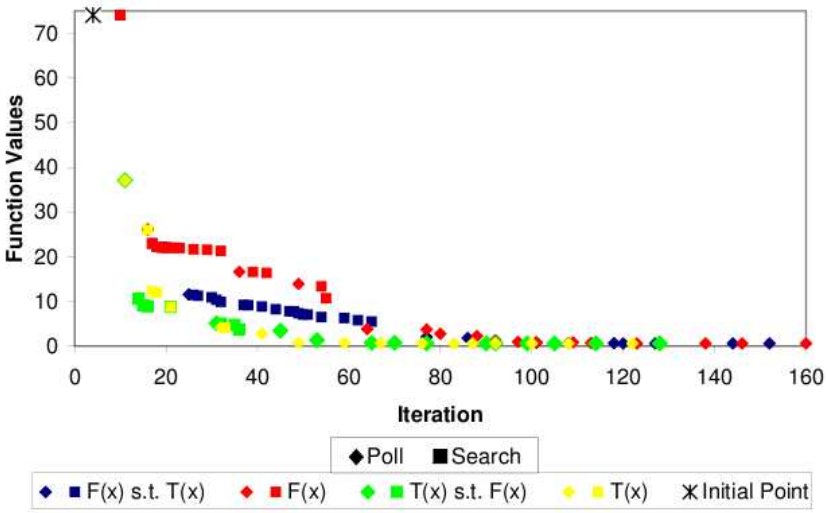

Fig. 4 Decreasing Function Value

function values appears to be a useful and efficient means of solving this class of problems. One challenge is dealing with the extent to which the CPU time correlation property holds in practice, which may not be fully understood. The implementation of the time cutoff parameter was a useful way to reduce the time required to find a numerical solution. However, while it can be used to stop the image registration algorithm, it cannot stop the numerical simulation, since the image registration requires the image obtained from the full simulation.

Since t_k^{cut} is controlled by the user, one potential improvement would be a more systematic approach to updating it, rather than the trial-and-error approach used here. Since function values and computational times are stored in order to construct surrogates, they can also be used to measure the correlation between the two. Higher values of t_k^{cut} can be assigned whenever the correlation is low, and vice versa.

Not using the surrogates when ill-conditioning occurs was a simple tactic that made sense for the particular problem we solved. However, a more effective means to combat this problem may be the use of a trust region (e.g., see [3]), both to constrain the optimization of the surrogate and to screen out points used in the construction of the surrogate. The size of the trust region could be based on the frame or mesh size parameter.

Acknowledgments

The authors wish to thank David Bethea for some useful comments and discussions.

The views expressed in this document are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or United States Government.

References

1. M. A. Abramson. NOMADm optimization software. http://www.afit.edu/en/ENC/-Faculty/MAbramson/NOMADm.html.
2. M. A. Abramson and C. Audet. Second-order convergence of mesh adaptive direct search. *SIAM J. Optim.*, 17(2):606–619, 2006.
3. N. Alexandrov, J. E. Dennis, Jr., R. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Structural Optim.*, 15:16–23, 1998.
4. C. Audet and J. E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13(3):889–903, 2003.
5. C. Audet and J. E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM J. Optim.*, 14(4):980–1010, 2004.
6. C. Audet and J. E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.*, 17(2):188–217, 2006.
7. A. J. Booker. Well-conditioned Kriging models for optimization of computer simulations. Technical Report M&CT-TECH-00-002, Boeing Computer Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124, 2000.
8. A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Struct. Optim.*, 17(1):1–13, February 1999.
9. M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics: a Practical Introduction*. SIAM, New York, 1998.
10. Jr. L. A. Schmit and H. Miura. Approximation concepts for efficient structural synthesis. Technical Report CR-2552, NASA, 1976.
11. R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM J. Optim.*, 9(4):1082–1099, 1999.
12. R. M. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. *SIAM J. Optim.*, 10(3):917–941, 2000.
13. R. M. Lewis and V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM J. Optim.*, 12(4):1075–1089, 2002.
14. S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the MATLAB toolbox DACE. Technical Report IMM-TR-2002-13, Technical University of Denmark, Copenhagen, 2002.
15. J. Modersitzki. *Numerical Methods for Image Registration*. Oxford University Press, 2004.
16. H. B. Nielsen. DACE surrogate models. http://www2.imm.dtu.dk hbn/dace.
17. A. B. Owen. Orthogonal arrays for computer experiments, integration, and visualization. *Statistica Sinica*, 2:439–452, 1992.
18. T. D. Robinson, M. S. Eldred, K. E. Willcox, and R. Haimes. Strategies for multifidelity optimization with variable dimernsional hierarchical models. In *Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (2nd AIAA Multidisciplinary Design Optimization Specialist Conference)*, Newport, Rhode Island, May 2006.
19. J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Stat. Sci.*, 4(4):409–435, 1989.
20. T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer Verlag, 2003.
21. M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.
22. Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993.
23. V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7(1):1–25, February 1997.